

作業の分担を考慮した Web アプリケーションの実装方針

Implementation Guidelines for Sharing Maintenance Tasks on Web Applications

芦田 昌也・竹内 昭浩・岩田 英朗
Ashida, Masaya & Takeuchi, Akihiro & Iwata, Hideaki

ABSTRACT

This paper reviews and discusses some implementation guidelines for sharing maintenance tasks on web applications using the relational database. These tasks are inclusive of page design modification and functionality enhancement, among others. These guidelines are conceived under the conditions that such web applications are based on a template engine and scripts coded using an object-oriented programming language. There are two guiding principles in sharing maintenance tasks. The first principle is the unconditional collection of data, which can accommodate cases of field additions and redundancies. The other principle lies in the concept of encapsulation which allows for a rational division of tasks between script programmers and page designers. Following these guidelines, the paper illustrates issues related to sharing maintenance tasks with the implementation of a simple web application.

1. はじめに

Web ブラウザで表示されたページをユーザインタフェースとし、Web サーバを介してサービスを提供するプログラムは、Web アプリケーションと呼ばれる。アプリケーション固有のソフトウェアを必要とせず、一般の Web ブラウザから

利用できることから、ネットワークで接続されたコンピュータに様々なアプリケーションを提供するひとつの形態となっている。

通常のアプリケーションソフトウェアと同様に、Web アプリケーションにおいても機能拡張や不具合修正などの保守作業は不可避である。保守作業にかかる作業コストを必要以上に増大させないようにするためには、(1) 作業内容の独立性を高めること、(2) 関連する実装箇所を局所化すること、などが考えられる。無計画に保守作業の担当分野を分割すると、分野間での調整を要することになるため、必要最小限のコミュニケーションで作業がすすめられるよう担当分野を決定することが望ましい。実装に関しては、ある部分での修正が他の部分に及ぼす影響を可能な限り小さくすることが望まれる。

Web アプリケーションの構築にあたっては、テンプレートエンジンの利用が提案されている。これは、Web アプリケーションの本体を成すスクリプトによる処理結果を、結果表示のための HTML 文書の原型であるテンプレートに動的に埋め込み、HTML 文書を生成するものである。この機能を用いると、テンプレートの記述とスクリプトの実装とを並行して実施することが可能になる。また、スクリプトの実装に際しては、オブジェクト指向言語を使用することにより、実装箇所の局所化を図ることが可能になる。

そこで本稿では、テンプレートエンジンとオブジェクト指向言語の利用を前提として、作業の独立性と実装の局所化を図る実装方針について述べる。対象とする Web アプリケーションは、リレーショナルデータベースから指定された条件に一致するデータを検索するデータ検索システムである。

テンプレートエンジンの利用に際しては、スクリプトとテンプレートの間で授受するデータが定められる。オブジェクトを利用してデータを授受することにより、スクリプトのプログラマとテンプレートのデザイナーの間で共有すべき知識を明確に限定できる。これにより、必要最小限のコミュニケーションにより、共同作業が進められるものと考えられる。

また、リレーショナルデータベースのテーブル定義と、オブジェクトのインス

タンス変数を連携させ、データの抽出などの処理に若干の冗長性を持たせておくことにより、テーブルの軽微な修正であれば、スクリプトへの影響を低減できるものと考えられる。

以下、2 章において対象とする Web アプリケーションの構成について述べる。3 章では、対象とする Web アプリケーションの構成要素ごとに、実装方針を検討・提案する。4 章では、提案した実装方針に基づいて、小規模の Web アプリケーションとして 2 種類のシラバス検索システムを構築し、提案する方針の有効性を検証する。5 章は、本稿のまとめであり、残された課題を示す。

2. Web アプリケーションの構成

本稿で対象とする Web アプリケーションは、すでにデータの格納を終えたりレージョナルデータベースから、指定された条件を満たすデータを抽出し、利用者に提示する一種のデータ検索システムである。利用者からのデータの挿入・更新・削除は行わないものとする。

2.1. 動作の概要

利用者からの要求に応じて、スクリプトとテンプレートエンジンは、検索条件を指定する HTML 文書を生成し、Web サーバを介してクライアントへ送信する。利用者は Web ブラウザを利用してその文書を閲覧し、検索条件を指定して、Web サーバへ返送する。スクリプトは、クライアントから送られた検索条件に基づいて対象となるレコードを選択し、抽出したデータをテンプレートに渡す。テンプレートエンジンは、渡されたデータをテンプレートの該当箇所で開催し、HTML 文書を生成する。生成された文書は、再度、Web サーバを介してクライアントへ送信され、利用者が結果を閲覧する。

2.2. テンプレートエンジン

テンプレートエンジンは、スクリプトの処理結果をテンプレートに埋め込み、

ページを動的に生成する。テンプレートには内容が動的に変化する部分に変数（以下、テンプレート変数）が埋め込まれる。スクリプトでは処理結果を変数（以下、ブリッジ変数）に代入し、テンプレート変数に割り当てる。テンプレートエンジンは、ブリッジ変数に代入されたデータを展開してテンプレート変数と置き換え、HTML 文書を生成する。

2.3. スクリプト

スクリプトは、Web アプリケーション全体の振る舞いをコントロールする。利用者からの要求の種類に応じて、適切なテンプレートを選択する。検索条件を満たすデータをリレーショナルデータベースから抽出し、ブリッジ変数に格納した上でテンプレート変数に割り当てる。

2.4. リレーショナルデータベース

リレーショナルデータベースには、通常、検索対象となる一連のデータを正規化した上で格納する。本稿では、検索対象となる一連のデータに第三正規化まで施す。これで得られるテーブルのうち、コードとそれに対応するラベルの2項からなるテーブルをラベルテーブルと呼ぶことにする。主キーを含むテーブルは、エントリテーブルとする。

3. 実装方針

3.1. テンプレートの記述

利用者が検索条件を指定するページ（以下、検索ページ）を生成するためのテンプレートと、検索結果を表示するページ（以下、表示ページ）を生成するためのテンプレートの2種類を記述する。

3.1.1. 検索ページテンプレート

検索ページでは、HTML の Form 要素を利用してスクリプトに検索条件を送

る。検索条件を指定する部分には、主に select 要素と input 要素を用いる。

方針 1 検索ページテンプレート

- I. 検索条件の指定に用いる要素の name 属性の値は、エントリテーブルにおけるフィールド名に一致させる
- II. ラベルテーブルが存在するフィールドを検索キーにする場合は、検索条件の指定方法を select 要素で選択式にする。この場合、
 - (ア) option 要素の記述に要するデータはブリッジ変数から取得する
 - (イ) ブリッジ変数から取得されるデータ数に依存しない方法で option 要素を記述する
 - (ウ) option 要素の value 属性の値は、ラベルテーブルのコードに一致させる
 - (エ) option 要素の内容は、value 属性の値に対応するラベルに一致させる
- III. ラベルテーブルが存在しないフィールドを検索キーにする場合は、検索条件を input 要素により入力させる。

Form 要素によってクライアントから送られるデータは、各要素の name 属性と value 属性の対である。方針 1 に従うことにより、スクリプトでは、SQL の WHERE 節を直接的に生成できる。また、方針 1- II -(ア)、および (イ) は、ラベルテーブルに修正が加えられた場合でも、検索ページテンプレートに追加的な作業を発生させないための方針である。

3.1.2. 表示ページテンプレート

表示ページでは、検索結果を利用者に提示する。

方針 2 表示ページテンプレート

- I. 表示に要するデータは、すべてブリッジ変数から取得する
- II. 検案件数に依存しない方法により、検索結果を記述する

スクリプトから渡されたブリッジ変数に含まれるデータの表示・非表示、および表示順序の変更である限り、その変更の影響がスクリプトとデータベースに及ばないようにするための方針である。

3.2. スクリプトの実装

3.2.1. ブリッジクラスの定義

テンプレート変数に割り当てるオブジェクトを生成するためのクラスをブリッジクラスと呼ぶことにする。

方針3 ブリッジクラス

I. 少なくとも次のインスタンス変数を定義する

- a) ラベルテーブルを格納するための連想配列で、その名前をラベルテーブル名に一致させたもの
- b) 選択された全レコードのデータを格納する連想配列の配列
- c) 上記以外に利用者に提示するデータを格納する変数または配列（連想配列を含む）

II. コンストラクタでは少なくとも次の処理を実行する

- a) すべてのラベルテーブルの内容を取得し、連想配列に格納する
- b) レコードを選択し、全レコードを配列に格納する。この配列の要素は、それぞれのレコードのデータを格納した連想配列とする
- c) 上記以外に利用者に提示するデータを変数または配列（連想配列を含む）に格納する

III. 少なくとも次のメソッドを実装する

- a) ラベルテーブル名を引数として、その内容をデータベースから取得して連想配列に返すメソッド
- b) エントリテーブルから、検索条件を満たすレコードのデータを連想配列に格納し、それを要素とする配列を返すメソッド

コンストラクタにおいて、すべてのインスタンス変数を適切に初期化する。データベースに変更が加えられた場合でも、このクラスに対して修正を加えるだけで対応を可能にする。

3.2.2. 条件節の生成

クライアントから送られた検索条件 $K_1 = v_1 \& K_2 = v_2 \& \dots \& K_m = v_m$ に基づいて、エントリテーブルから該当するレコードを選択するための条件節を生成する。

方針 4 条件節の生成

次のいずれかの方法により生成された条件 C_1, C_2, \dots, C_m から生成される節 $C_1 \text{ AND } C_2 \text{ AND } \dots \text{ AND } C_m$ を条件節とする

(ア) 検索キー K_i の値が単一の値 v_i である場合, $K_i = v_i$ を検索キー K_i の条件 C_i とする

(イ) 検索キー K_j の値が複数の値 $v_j[0], v_j[1], \dots, v_j[n]$ である場合,
($K_j = v_j[0]$) OR ($K_j = v_j[1]$) OR ... OR ($K_j = v_j[n]$) を検索キー K_j による条件 C_j とする

この方針により、検索キーの追加・変更が行われた場合でも、条件節の生成処理は影響を受けなくなる。

3.2.3. レコードの選択

生成された条件節を用いて、エントリテーブルから検索条件に該当するレコードを選択する。

方針 5 レコードの選択

I. レコードを選択する際には、すべてのフィールド f_i ($i=1, \dots, n$) のデータ v_i ($i=1, \dots, n$) を抽出する。

II. 抽出したデータは、レコードごとにフィールド名 f_i をキーとする連想配列 $F_j[f_i] = v_1, F_j[f_2] = v_2, \dots, F_j[f_n] = v_n$ に格納する。

エントリテーブルに新規フィールドが追加された場合でも、スクリプトのこの部分への修正が生じないようにするための方針である。

3.3. コードの定義

本稿が対象とするシステムでは、あらたなレコードの追加が行われないことを前提としている。しかし、将来の拡張として、新規レコードが追加される場合でも、既存の文字列データのいずれかになることが確定しているフィールドについては、その文字列データに対応するコードを定義する。

方針 6 コードの定義

文字列データを格納するフィールドのうち、新規レコードを追加した場合でもあらたな文字列データの追加が生じないフィールドについては、そこに格納される文字列データに数値コードを定義する。

4. シラバス検索システムへの適用

提案した実装方針に従い、本学の基礎教育科目と本学部の専門教育科目の2種類のシラバスを対象にシラバス検索システムを構築する。

4.1. 実装環境

次のような環境においてシラバス検索システムを構築する。

テンプレートエンジン	: Smarty
スクリプト記述言語	: PHP (PHP : Hypertext Preprocessor)
リレーショナルデータベース	: PostgreSQL
Web サーバ	: Apache

4.2. データ構成

基礎教育科目・専門教育科目の両方に共通のエントリテーブルとする。エントリテーブルを構成するフィールドは次のとおりである。太字で表記したフィールドは、ラベルテーブルを有することを表す。

データ管理用フィールド

科目コード・ユーザ ID・科目区分・科目分野・科目種類

データ表示用フィールド

科目名・科目名（英語表記）・単位数・授業形態

対象学部・対象学科・対象学年

開講期間・授業曜日・授業時間

担当教員・オフィスアワー

授業概要・授業計画・成績評価・教科書・参考書・履修上の注意

4.3. 検索キー

基礎教育科目と専門教育科目のそれぞれについて、シラバスの検索キーとなるフィールドを次のように定める。

基礎教育科目

開講期間・授業曜日・授業時間

対象学部・対象学年

科目分野

担当教員名

科目名

専門教育科目

開講期間・授業曜日・授業時間

対象学科・対象学年

担当教員名

科目名

科目名については、入力語句が科目名に含まれる場合に、一致するデータが存在するものとみなす。科目名以外の項目は、リストから値を選択させる。

4.4. ブリッジクラスの定義

4.4.1. 共通クラス

基礎教育科目と専門教育科目の両方に共通するラベルテーブル

科目区分・授業形態・対象学年・開講期間・授業曜日・授業時間を格納する連想配列を定義する（方針 3-I-a）。また、検索条件として一覧から選択させるために必要となる、担当教員の氏名を格納するための連想配列を定義する（方針 3-I-c）。

4.4.2. 基礎教育科目サブクラス

共通クラスを継承し、基礎教育科目サブクラスを定義する。このサブクラスでは、新たに、科目分野と対象学部ラベルテーブルを格納する連想配列を定義する。これにより、基礎教育科目で使用するすべてのラベルテーブルを連想配列に格納できる。コンストラクタにおいて、基礎教育科目の全レコードから、担当教員の氏名を重複なく抽出し、担当教員名の連想配列に格納する（方針 3-II-c）。また、このサブクラスにおいて、基礎教育科目に特化したレコード抽出メソッドを定義する（方針 3-III-b）。

4.4.3. 専門教育科目サブクラス

基礎教育科目サブクラスと同様に、共通クラスを継承し、専門教育科目サブクラスを定義する。このサブクラスでは、新たに科目種類と対象学科のラベルテーブルを格納する連想配列を定義する。これにより、専門教育科目で使用するすべてのラベルテーブルを連想配列に格納できる。コンストラクタにおいても同様に、専門教育科目の全レコードから、担当教員の氏名を抽出し連想配列に格納する（方針 3-II-c）。また、このサブクラスにおいて、専門教育科目に特化したレコード抽出メソッドを定義する（方針 3-III-b）。

4.5. スクリプトの概要

2種類のスクリプトで構成される。ひとつは、検索キーの値を設定させるページと、指定された検索条件を満たす科目の一覧を表示するためのスクリプトである。このスクリプトの概要を図1に示す。searchpage.tmplは、検索キーの値

```
$templateObject = new TemplateEngin();  
$bridge = new BridgeObject();  
if ( isset( $searchCondition ) ) {  
    $templateFile = 'result.tmpl';  
    $bridge->setRecord( $searchConditions );  
} else {  
    $templateFile = '/path/to/searchpage.tmpl';  
}  
$templateObject->assignVariable( 'syabs', $bridge );  
$templateObject->displayTemplate( $templateFile );
```

図 1 検索・一覧画面用スクリプト

を指定させるページのテンプレートファイル名である。また, result.tmpl は, 検索結果の一覧を表示するページのテンプレートファイル名である。使用される変数名・メソッド名と, その意味を表 1 に示す。

もうひとつのスクリプトは, 検索結果である科目の一覧から選択された科目のシラバスを表示するためのスクリプトである。図 2 にその概要を示す。syllabus.tmpl はシラバスを表示するためのテンプレートのファイル名である。\$subjectCodes は, 一覧から選択された科目コードの配列である。科目コードは, エントリテーブル中のフィールドのひとつであるから, 検索キーの値と同様の処理が可能である。それゆえ, setRecord メソッドを使用できる。\$subjectCodes が定義されない場合は原則としてこのスクリプトを起動することがないが, フェールセーフとして検索キーを指定させるページを表示するものとする。

表1 変数・メソッドの意味

変数名	意味
\$templateObject	テンプレートエンジンのオブジェクト
\$bridge	ブリッジクラスのオブジェクト
\$searchCondition	サーバから送られた検索キー列
\$templateFile	テンプレートファイル名

メソッド名	意味
TemplateEngine()	テンプレートクラスのコンストラクタ
BridgeObject()	ブリッジクラスのコンストラクタ
setRecord()	エントリテーブルからレコードを抽出
assignVariable()	テンプレート変数への割り当て
displayTemplate()	テンプレートから HTML 文書を生成

```

$templateObject = new TemplateEngin();
$bridge = new BridgeObject();
if ( isset( $subjectCodes )) {
    $templateFile = 'syllabus.tpl';
    $bridge->setRecord( $subjectCodes );
} else {
    $templateFile = '/path/to/searchpage.tpl';
}
$templateObject->assignVariable( 'syabs', $bridge );
$templateObject->displayTemplate( $templateFile );

```

図2 シラバス表示用スクリプト

4.6. 動作例

基礎教育科目のシラバス検索システムの動作例を示す。ページにアクセスすると、検索条件を指定する画面を表示する（図 3）。検索条件が指定されると、その条件を満たす科目の一覧を表示する（図 4）。その一覧から、閲覧したい科目が選択されると、その科目のシラバスを表示する（図 5）。

基礎教育科目 シラバス検索

使い方

- 条件を指定して、検索ボタンを押してください。条件に一致する授業科目の一覧が表示されます。
- 授業時間の欄は、左から順番に開講期間(前期・後期など)・曜日・時限です。
- 条件を何も指定せずに検索ボタンを押すと、登録されているすべての授業科目の一覧が表示されます。

利用上の注意

- 表示されない授業科目もあります。
- 内容については、
 - 配布された時間割
 - 配布された用子体のシラバス
 - 指示
 を必ず確認してください。

検索

授業時間 指定なし 指定なし 指定なし

対象学部 指定なし

対象学年 指定なし

科目名 科目名の一部に を含む

担当教員名 指定なし

分野 指定なし

検索

ページが表示されました

インターネット

図 3 検索条件の指定画面



図 4 対象科目の一覧画面

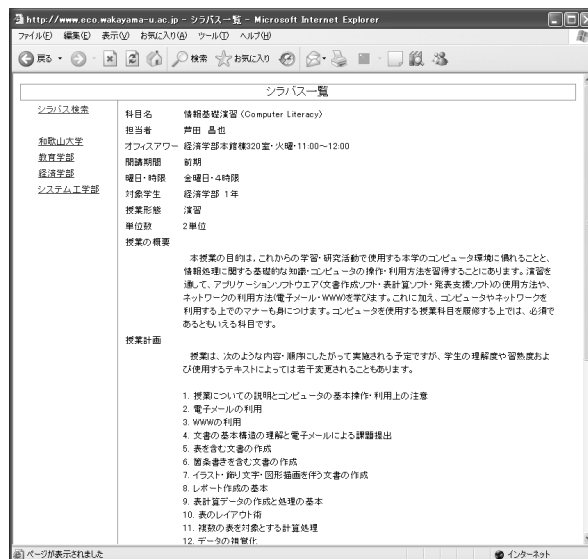


図 5 シラバス表示画面

4.7. 類似システムの追加構築

基礎教育科目シラバス検索システムを拡張し、専門教育科目シラバス検索システムを構築する。本稿では、両システムの構築を前提にエントリテーブルやクラスを定義しているが、実際には、いずれか一方のみを構築した後に、もう一方を追加構築するような状況が想定される。ここでは、そのような状況を想定し、実装方針の有効性を検証する。

4.7.1. データベースの拡張

基礎教育科目と専門教育科目のシラバスでは、掲載項目がやや異なる。そのため、新たにエントリテーブルを追加するか、または既存のエントリテーブルに不足項目を追加する必要がある。いずれの場合においても、ラベルテーブルの追加作業が生じる。

4.7.2. ブリッジクラスの拡張

前記のようなデータベースの拡張に伴い、ブリッジクラスの拡張が必要となる。3章で示したようなクラスの階層構造に再定義するか、または一方のシステムに特化して定義されたクラスを継承して、他方のクラスをサブクラスとして定義するかのいずれかである。いずれの場合についても、コンストラクタにおいて、追加されたラベルテーブルを取得する処理が必要となる。

4.7.3. スクリプトの変更

スクリプトについては、生成するブリッジオブジェクトを再定義（もしくはサブクラスとして継承）したクラスから生成するよう変更する。

4.7.4. テンプレートの変更

テンプレートでは、表示項目の追加・削除と検索キーとして使用する項目の追加・削除を行う。この際には、エントリテーブルに追加されたフィールド名

を把握しておく必要がある。

4.8. 実装方針の有効性

4.8.1. 実装の局所化

類似システムへの拡張が行われる場合、事前にデータベースへの追加項目などを想定していない限り、データベースの拡張作業は不可避である。

提案した実装方針では、レコードのすべてのフィールドの値を取得するというやや冗長な処理であるが（方針5-I）、これに従うことにより、エントリテーブルに項目の追加が行われた場合でも、レコードの選択メソッドの変更は不要である。また、検索キーの変更を行う場合でも、方針4に従うことにより、サーバから送られた検索条件からレコード選択のための条件節（WHERE節）への変換操作に新たな変更は生じない。したがって、ブリッジクラスに関しては、コンストラクタの変更に限定できる。

以上から、提案した実装方針により、特に作業コストを増加させることなく、実装を局所化することが可能であると考えられる。

4.8.2. 作業の分散と情報共有

スクリプトの変更に際しては、データベースの拡張に伴い、再定義されたクラスに関する情報を把握すればよい。新規構築時も含め、保守作業を行う際には、必然的に使用するクラスの仕様を把握する必要がある。したがって、再定義されたクラスに関する情報共有のための作業は、提案した実装方針のために追加的に必要性が生じるものとはならない。

テンプレートの変更に際しては、データベースに追加されたラベルテーブルと、エントリテーブルに追加されたフィールド名を把握する必要がある。新規のインスタンス変数としてラベルテーブルが定義されるが、データベースのラベルテーブル名とその変数名を一致させる限り、テンプレートの変更時にクラス定義を参照する必要は生じない。ブリッジオブジェクトを介さない場合でも、

それぞれのテンプレート変数に割り当てられる値を把握する必要があることから、データベースに追加されたラベルテーブル名やフィールド名を把握することが、提案した実装方針に従うことにより生じる追加的な作業とはならない。

以上から、提案した実装方針に基づくことにより、作業コストを増加させることなく、作業の分散を図ることが可能になると考えられる。

5. おわりに

新たな作業コストを生じさせることなく、類似システムへの拡張時に作業を分散させることと、実装の変更箇所を局所化することを目的とした実装方針を提案した。対象となるシステムは、リレーショナルデータベースに格納されたデータを抽出する Web アプリケーションである。本稿では、利用者によるデータの更新機能を取り扱っていない。しかし、特定のフィールドに固有の処理を必要としない限り、エントリテーブルのフィールド名と Form 要素の name 属性の値を一致させるという実装方針に従えば、データ更新機能を追加することも容易である。

本稿では、ブリッジ変数にオブジェクトを使用した。必要なデータにアクセスするには、テンプレート内でインスタンス変数名や、データベースに格納されている各種テーブルのフィールド名を指定する必要がある。インスタンス変数名の変更やフィールド名の変更を想定すると、オブジェクトを使用しない場合には、スクリプトにおいて変更点を吸収することが可能である。しかし、オブジェクトを使用する方法では、テンプレートでの変更も必要となる。そのためブリッジクラスにおいて、インスタンス変数名やフィールド名を隠蔽する一種のアクセスを定義し、テンプレートでの変更が不要となるように工夫を要する。通常、シラバスのようなデータを格納するには、リレーショナルデータベースより XML データベースを使用するか、またはシラバスを記載した XML ファイルに付した属性情報のみをリレーショナルデータベースに格納するなどの方法がとられる。このような形態への移行を検討することも今後の課題である。

参考文献

山田祥寛, Smarty 入門, 翔泳社 (2005)

大垣靖男, はじめての PHP 言語プログラミング入門, 技術評論社 (2005)

鶴田展之, 繁田卓二, PHP 実用プログラミング, ローカス (2004)

鈴木啓修, PostgreSQL 全機能リファレンス, 技術評論社 (2003)

山田祥寛, PEAR 入門, 翔泳社 (2005)

東海林誠, 最新オブジェクト指向開発マスタリングハンドブック, 秀和システム (2004)

Matt Weisfeld, 萩本順三 (訳), Java と UML で学ぶオブジェクト指向の考え方, 翔泳社 (2002)

磯田定宏, オブジェクト指向モデリング, コロナ社 (2000)